Please type a plus sign (+) inside this box →

☐ +

UTILITY PATENT APPLICATION TRANSMITTAL (Only for new nonprovisional applications under 37 CFR 1.53(b))	*Attorney Docket No.*	C99021US
	First Named Inventor or Application identifier	**MARY ELLEN ZURKO**
ADDRESS TO: Assistant Commissioner of Patents Box Patent Application Washington, DC 20231	*Title*	VERIFICATION OF TRUSTED-PATH COMMANDS
	Express Mail No.	EL357886594US

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents

1. _X_ Fee Transmittal Form PTO/SB/17 *(submit in duplicate)*
2. _X_ Specification [Total Pages: _67_]
3. _X_ Drawing(s) (35 USC 113) [Total Sheets: _7_]
4. _X_ Oath or Declaration [Total Pages: _2_]
   a. __ Newly executed (original or copy)
   b. _X_ Copy from a prior application (37 CFR 1.63(d))
      (for continuation/divisional with No. 17 completed)
      [Note No. 5 below]
      i. __ DELETION OF INVENTOR(S)
         Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).
5. _X_ Incorporation By Reference *(useable if No. 4b is checked)*
   The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under No 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
6. __ Microfiche Computer Program *(Appendix)*
7. __ Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
   a. __ Computer Readable Copy
   b. __ Paper Copy (identical to computer copy)
   c. __ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. __ Assignment Papers (cover sheet & document(s)
9. __ Power of Attorney
10. __ English Translation Document *(if applicable)*
11. __ Information Disclosure Statement (IDS)/PTO-1449
    __ Copies of IDS Citations
12. __ Preliminary Amendment
13. _X_ Return Receipt Postcard *(Itemized)*
14. __ Small Entity Statement(s)
    __ Statement filed in prior application. Status still proper and desired
15. __ Certified Copy of Priority Document(s) *(if foreign priority is claimed)*
16. __ Other: _____

17. If a CONTINUING APPLICATION, check appropriate blank and supply the requisite information:
    __ Continuation
    _X_ Divisional   of prior application No.: _07/479,666_
    __ Continuation-in-part (CIP)

18. Correspondence Address				
X Customer Number or Bar Code Label _001200_ (Insert Customer No. or Attach bar code label here)		or	X Correspondence address below	
Name	AKIN, GUMP, STRAUSS, HAUER & FELD, L.L.P.			
Address	711 Louisiana, Suite 1900			
City	Houston	*State*	Texas	*Zip Code* 77002
Country	U.S.A.	*Telephone* (713) 220-5800	*Fax*	(713) 236-0822

David R. Clonts, Reg. No. 36,768

Date: 2/29/00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Applicants: §
§
    MARY ELLEN ZURKO §
    THOMAS A. CASEY, JR. §
    MORRIE GASSER §
    JUDITH S. HALL §
    CLIFFORD E. KAHN §
    ANDREW H. MASON §
    PAUL DOUGLAS SAWYER §
    LESLIE RICHARD KENDALL §
    STEVEN B. LIPNER §
§
Filed: Concurrently Herewith §   Art Unit:
§
Serial No.: Unknown §   Examiner:
§
For: VERIFICATION OF TRUSTED- §   Docket No.: C99021US
    PATH COMMANDS §

PRELIMINARY AMENDMENT

BOX PATENT APPLICATION
Assistant Commissioner for Patents
Washington, DC 20231

Dear Sir:

Please amend the divisional application as follows before consideration of this case:

In the Title:

Please replace the current title with the new title "VERIFICATION OF TRUSTED-PATH COMMANDS".

In the Claims:

Please cancel claims 1-20.

Please add new claims 21-28:

1   21.   (New) A machine-executed method for verifying the existence of a trusted

2 path to a user in a computing system, the computer system including a trusted computing

3 environment, the method comprising the steps of:

4   (a)   upon login by a user, assigning a process identifier to the user in the

5 trusted computing environment;

6   (b)   storing the assigned process identifier in trusted memory;

7   (c)   establishing a trusted path between the user and the trusted computing

8 environment;

9   (d)   through the trusted path, displaying the process identifier to the user;

10 and

11   (e)   upon the user's subsequent entry into the trusted environment,

12 displaying the process identifier to the user through the trusted path.

1   22.   (New) The method of claim 21, wherein the process identifier is a randomly

2 or pseudo-randomly generated group of alphanumeric characters.

1   23.   (New) The method of claim 21, wherein the process identifier is

2 pronounceable.

1   24.   (New) An automatic data processing machine programmed to execute the

2 method of any one of claims 21 to 23.

1   25.   (New) An automatic data processing machine comprising means for

2 performing the method steps of any one of claims 21 to 23.

1   26.   (New) A program storage device readable by a machine and tangibly

2 embodying a representation of a program of instructions adaptable to be executed by said

3 machine to perform the method of any one of claims 21 to 23.

1   27.   (New) Apparatus for executing a trusted command that is issued by a user and

2 that is parsed by untrusted parsing means to generate a parsed command, comprising:

3   (a)   trusted means for receiving the parsed command via a trusted path;

4   (b)   means for displaying a representation of the parsed command to the

5 user for verification; and

6   (c)   trusted means for executing the verified parsed command.

2

1   28.   (New) A program storage device readable by a machine and encoding
2   machine-executable instructions that include instructions for performing the methods of any
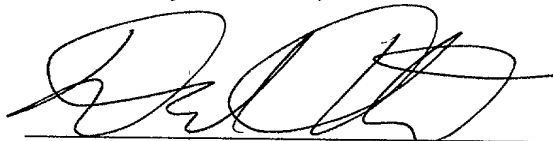3   one of claims 21 through 23.

REMARKS

This Preliminary Amendment presents new claims 21-28 which are directed to verification of trusted-path commands. Claims 21-28 essentially correspond to non-elected claims 10-16 and 21 from the parent case. Unlike the non-elected claims 13-15 and 21 in the parent case, new claims 24-28 depend only from the claims in this case.

It is respectfully submitted that the present application is in condition for allowance and such actions are respectfully requested. If there are any questions or comments regarding this Preliminary Amendment, the Examiner is encouraged to contact George W. Jordan III or David R. Clonts at (713) 220-5800.

Respectfully submitted,

_2/29/99_
Date

David R. Clonts, Reg. No. 36,768

AKIN, GUMP, STRAUSS, HAUER & FELD, L.L.P.
711 Louisiana, Suite 1900
Houston, TX 77002
(713) 220-5800
(713) 236-0822 Fax

METHOD FOR EXECUTING TRUSTED-PATH COMMANDS

BY

Mary Ellen Zurko
Thomas A. Casey, Jr.
Morrie Gasser
Judith S. Hall
Clifford E. Kahn
Andrew H. Mason
Paul Douglas Sawyer
Leslie Richard Kendall
Steven B. Lipner

TABLE OF CONTENTS

1. BACKGROUND OF THE INVENTION

1.1 Introduction

This invention relates to a method for improving security in a computer system. More specifically, this invention concerns a method for implementing trusted commands through both trusted and untrusted code.

1.2 Background

The proliferation of computers has resulted in the need for reliable computer security systems. The need to protect certain information is great in areas dealing with national security, for example. Security measures are required in other fields, including areas which utilize financial, medical, and personal information. Many computerized information systems therefore include internal security measures which provide the users with different levels of access to the computerized information, and which attempt to provide at least a degree of protection of the information from undesirable access.

1   1.3   <u>Security Criteria; Reference Validation Systems</u>

2

3   In response to the need for secure computer systems, the

4   Department of Defense has issued a publication titled <u>Department of</u>

5   <u>Defense Trusted Computer System Evaluation Criteria</u> (reference No.

6   DOD 5200.28-STD). This publication is sometimes referred to as the

7   "Orange Book," and is available from the Department of Defense. The

8   Orange Book describes system security objectives and evaluation criteria

9   for secure computer systems.

10

11   A "secure" computer system generally includes some type of

12   "reference validation" system. These reference validation systems (known

13   as reference monitors) are responsible for enforcing the security rules

14   (security policy) for a given computer system.

15

16   Reference monitors mediate all access to "objects" by "subjects".

17   Objects are passive entities that contain or receive information. Access

18   to an object implies access to the information that it contains. Subjects

19   are active entities, generally in the form of a person, process or device

20   that cause information to flow among objects or change the system state.

21   A subject may ordinarily reference its own, subject-internal information

22   without the involvement of the reference monitor. Reference monitors

23   controlling such access to objects by subjects are known in the art, and

24   may utilize a security kernel approach.

1   Proper implementation of a reference monitor calls for adherence

2   to three principles:

3

4   (1) completeness, in that all access by

5   subjects to objects or other subjects must

6   involve the monitor;

7

8   (2) isolation, in that the monitor must

9   be protected from tampering; and

10

11   (3) verifiability, in that correspondence

12   must be shown between the security policy

13   and the actual implementation of the monitor.

14

15   As discussed, every reference to information or change of

16   authorization should go through the reference monitor. Thus, all

17   commands issued by a user or other subject are monitored by the

18   reference monitor. This approach is particularly useful in multiuser

19   computer environments.

20

21   The totality of the protection mechanisms within a computer

22   system -- including hardware, software, and firmware, the combination of

23   which is responsible for enforcing a security policy -- is commonly

24   known as a "trusted computing base" (TCB). If the trusted software is

3

1   designed to be as simple as possible, for the sake of verifying the

2   reference monitor, then the trusted software is known as a "security

3   kernel".

4

5   Generally, TCBs attempt to meet the control objectives set out in

6   the Orange Book. Compliance with these objectives inspires user

7   confidence, and increases the overall desirability of a computer system.

8   These objectives deal with:

9

10   (1) security policy;

11   (2) accountability; and

12   (3) assurance.

13

14   The security policy objective entails enforcement by the TCB of

15 the desired security rules. These security rules are designed to limit the

16 access to and dissemination of information in a precisely defined

17 manner.

18

19   Security policies may include provisions for the enforcement of

20 both mandatory and discretionary access control rules.   Mandatory

21 access control rules control access based directly on comparisons of the

22 user's security level and the sensitivity level of the information being

23 sought. Discretionary access rules control and limit access to identified

24 · individuals who have been determined to have a need-to-know.

1

2   These access control rules call for associating with each user

3 identification code a statement indicating the user's access rights. This

4 statement often includes information representing the user's security level

5 (for mandatory control purposes), and membership in groups (for

6 discretionary control purposes).

7

8   The accountability objective calls for providing each user with an

9 individual user identification code (often called a "user name") and for

10 the TCB to be able to recognize the code and ensure that it is being

11 used by its proper user. This may be done by checking the user's

12 password. This ensuring the user's identity is known as "authentication."

3

14   In addition, the accountability requirement calls for the existence

15 of auditing capabilities. Such capabilities allow for the auditing of

16 actions which can cause access to, generation of, or release of classified

17 or sensitive information.

18

19 1.4   <u>Assurance Objectives and "Trusted" Systems</u>

20

21   The assurance objective is especially important in the present

22 context. That objective is concerned with taking steps to ensure that the

23 security policy is correctly implemented and that the TCB accurately

24   mediates and enforces the intent of that policy. Steps may be taken to

1   insure that each portion of the TCB is assured. To accomplish this

2   objective, two types of assurance are needed.

3

4   The first type of assurance is life-cycle assurance. This type of

5   assurance refers to steps taken to ensure that the computer system is

6   designed, developed, and maintained using formalized and rigorous

7   control standards.

8

9   The second type of assurance is operational assurance.

10   Operational assurance deals with the features and system architecture

11   used to ensure that the security policy is uncircumventably enforced

12   during system operation. All of the software (sometimes referred to

3   informally in the art as "code") in the TCB is generally analyzed to

14   determine the assurance level of the system.

15

16   As the amount of code in the TCB increases, it becomes more

17   difficult to ensure that the TCB accurately enforces the security policy.

18   Because of this, it is desirable to minimize the amount of trusted code,

19   and thus the complexity of the TCB.

20

21   A TCB is usually operated with a substantial amount of software,

22   such as text editors and other applications, operating within the security

23   policy of the TCB. Generally, this untrusted software asks the TCB for

24   access to objects when the user or the untrusted software requires them.

1   Thus, the majority of user's requests to the TCB, and the majority of the

2   information that a user obtains from the TCB, are handled through the

3   agency of untrusted software.

4

5   This untrusted software, however, is by nature in danger of

6   compromise and vulnerable to bugs. For some types of requests and

7   displays, malicious or malfunctioning untrusted software could

8   compromise the enforcement of the security policy. Generally, TCBs

9   cannot distinguish between requests faithfully made by the untrusted

10 software on command from a user and either requests made by the

11 untrusted software at its own initiative or requests that misrepresent the

12 user's actual command. For example, if commands issued by an

3   authorized user to change certain users' security levels were made

14 through the agency of untrusted software, it would be possible for

15 malicious or malfunctioning untrusted software to inappropriately raise

16 the security level of a user. Such inappropriate raising of a security

17 level could result in the disclosure of sensitive information.

18

19   Furthermore, TCBs generally cannot ensure that displays made by

20 untrusted software are faithful. This poses problems in that if displays

21 of audit records were made through the use of untrusted software it

22 would be possible for malicious untrusted software to misrepresent these

23 audit records to hide suspicious activities.

24

1   To overcome these problems, prior-art systems have developed a

2   concept known as a "trusted path." A trusted path is a mechanism by

3   which the user can communicate directly with the TCB. A trusted path

4   may only be activated by the user or the TCB and cannot be imitated

5   by untrusted code. Such a trusted path is a positive TCB-to-user

6   connection that bypasses all untrusted software. The Orange Book

7   requires a trusted path for all systems to have a security rank of B2 or

8   above.

9

10   A "trusted command" (also known as a trusted-path command) is

11   a command which requires a trusted path between the user and the TCB

12   for execution. The specific security policy of a computer system will

3   determine which commands are defined as trusted commands. For

14   example, commands relating to changes of the security levels of users

15   would be trusted commands.

16

17   1.5   Parsers as Increasers of System Complexity

18

19   Because of perceived performance problems, prior-art computer

20   systems often included code to implement certain functions in the TCB.

21   One such function comprises a portion of code known as the "parser."

22

23   The parser performs the basic interpretation of user commands by

24   translating a human-readable representation of a user command (e.g., a

command typed by a user at a keyboard) into a machine-readable representation known as the binary representation, or the parsed command. A user command may consist of a group of words, typed by the user, specifying some action to be taken. The user command may also specify one or more variations or options of that action.

For the convenience of the user, most parsers allow these option words to be typed in any order without changing the meaning of the user command. Also, most parsers allow the user to type the words in either full or abbreviated form. In some instances, several possible abbreviations exist. Further, in most parsers, the user may vary the spacing between the words without changing the meaning of the command. Some parsers also check user commands for proper syntax, and report errors to the user prior to execution of the command. Other parsers prompt the user for missing information, avoiding the need to completely retype an incomplete command.

Thus, several different inputted human-readable representations may be associated with each unique parsed command. It is the job of the parser to translate accurately these inputted representations of user commands (which may vary with each user) into specific parsed command representations. Due to the complex nature of parsing, large amounts of computer code are generally associated with this activity.

1   Prior-art trusted software systems have included the parser for

2   trusted commands, and thus the associated code, within the TCB. In

3   these systems every trusted command was parsed and executed

4   exclusively by trusted code. The inclusion of the parsing code in the

5   TCB was regarded as necessary for proper system operation.

6

7   As discussed above, however, the ease and degree of assuring a

8   system is roughly inversely proportional to the complexity of the system's

9   TCB. Thus, the inclusion of the parser in the TCB results in more

10   complex assurance analysis and a corresponding decrease in system

11   confidence. Prior art devices (since they needed to be small and simple

12   to support assurance) have thereby inherently constrained the user-

'3   friendliness of computer systems.

14

15   ## 2. SUMMARY OF THE INVENTION

16

17   The present invention provides a number of advantages over the

18   prior art. By reducing the complexity of the TCB, the amount of trusted

19   code that must be verified or assured is reduced. In addition, the

20   general usability of the system is increased, because the improved

21   method of processing trusted commands allows for the implementation of

22   added computing functions at little or no additional cost in TCB

23   assurance.

24

1   The present invention includes an improved method for the

2   execution of trusted commands. In this improved method, the required

3   steps for processing a trusted command are split between trusted and

4   untrusted code. In this manner the amount of trusted code in the TCB

5   may be significantly reduced.

6

7   As discussed above, the parsing of a command can be compara-

8   tively difficult, and often requires large amounts of complex code.

9   However, it is relatively simple to translate the parsed representation of

10   a command into a standard human-readable representation for display to

11   the user. This is in part because a single, standard human-readable

12   display representation may be associated with each parsed representation.

3   The present invention takes advantage of this fact by parsing the trusted

14   commands in untrusted code prior to execution in the TCB.

15

16   In general, the present invention includes a computer system

17   which includes a TCB operating as a reference monitor. Included in the

18   TCB is the security-relevant code necessary for the actual execution of a

19   trusted command.

20

21   Associated with the TCB are one or more untrusted subjects (e.g.

22   processes). Each such subject has particular access rights, which may

23   include a security level. These untrusted subjects communicate only with

24   the involvement of the TCB. The TCB allows communication only as

1   permitted by the system's security policy. Each untrusted subject is

2   substantially independent.

3

4   In each of these untrusted subjects, untrusted software is generally

5   run. This untrusted software may include an untrusted operating system

6   (UOS), and generally includes applications such as text editors. In the

7   present invention, the untrusted code includes the non-security relevant

8   functions necessary for the execution of a trusted command.

9

10   Attached to the computer system are several user terminals.

11   These user terminals allow the user to exchange information with the

12   computer system. All terminal activity is seen first by the TCB. A

13   secure-attention key is dedicated for use on each terminal attached to

14   the computer system. The TCB is designed to recognize and trap this

15   key signal. Activation of this key by the user establishes a trusted path

16   between the user terminal and the TCB. Alternate methods for

17   establishing a trusted path may be utilized.

18

19   The TCB is generally provided with a limited number of

20   "direct" user commands (direct commands). Because of this, the majority

21   of the user's activities will take place in the untrusted operating system.

22

23   While operating in the UOS, the user may attempt to execute a

24   trusted command. When one of these commands is issued by the user,

12

1   the untrusted code in the UOS checks the syntax of the command and

2   may prompt for missing parameters; then the untrusted parser converts

3   the command into a parsed representation that will be referred to here

4   for convenience as the "binary representation." In some instances,

5   command procedures and application-issued trusted commands will also

6   be parsed and translated into a binary representation.

7

8   This binary representation is then passed from the UOS to the

9   TCB. The TCB initially associates the binary representation with a

10   physical terminal, and copies the binary representation into a memory

11   buffer assigned to that terminal.

12

3   Once this has been completed, the untrusted code in the UOS

14   may prompt the user to activate the secure-attention key. If this key is

15   activated, a trusted path will be established between the user terminal

16   and the TCB. Alternate methods may be employed to initiate this

17   trusted path. A randomly generated process ID, which may be

18   generated at login, may be utilized to prevent untrusted code from

19   simulating a trusted path.

20

21   Subsequent activity by the TCB depends on the type of command

22   that is represented by the binary representation. If the command is one

23   that will not modify any information, i.e., a request to view information,

24   the TCB verifies that the user has the necessary access rights, and

1 attempts to execute the binary representation. A human-readable

2 standard representation of the binary command received by the TCB is

3 displayed along with the information.

4

5 In most instances, if the command requested by the user requests

6 modification of information, the TCB displays for confirmation a

7 standard human-readable form of the command, along with an indication

8 of what the result of command execution will be. The user is then

9 requested to confirm (by typing a response via the trusted path) that the

10 displayed command was received correctly and should be executed.

11

12 This confirmation allows the user to ensure that untrusted code

3 has not performed an unauthorized modification of the user's original

14 command. Also, confirmation allows for the detection by the user of

15 any errors that may have occurred during the parsing process. In

16 situations where the binary representation represents a command

17 procedure or an application-issued trusted command, the confirmation

18 ensures that the command proposed by the untrusted software is

19 acceptable to the user.

20

21 If the command is confirmed by the user, the TCB checks to

22 ensure that the user has the necessary access rights to execute the

23 command and, if so, attempts to execute the command.

24

1   A distinct advantage of the present invention arises from the fact

2 that the execution of a trusted command is split between trusted code

3 and untrusted code. The actual parsing of the command occurs in

4 untrusted code. The trusted code included in the TCB checks the

5 parsed command for correctness and displays it to the user in human

6 readable form. This checking and display of a parsed command requires

7 substantially less code than is required to parse a command.

8

9   Thus, since the command parser is included in the untrusted code,

10 the amount of trusted code in the TCB is minimized. This lowers the

11 difficulty of trusted system assurance, and increases the amount of

12 confidence that may be placed in a system.

3

14   It is a further advantage of this invention that the user interface

15 is separate from the underlying functions performed by the TCB. This

16 allows for implementation of alternate UOS-to-user interfaces without

17 major modification of the TCB. Further, the amount of time and effort

18 to develop such a command interface is reduced. A still further

19 advantage is that more useability features may be provided without

20 substantially increasing the complexity of the TCB. Examples of such

21 useability features are command line editing and recall.

22

Figure 1 illustrates the various levels within a computer system;

Figure 2 illustrates various layers within the trusted computing base;

Figure 3 illustrates a trusted path;

Figures 4A and 4B are flow diagrams representing a method for processing trusted commands. Differences in box representation in these drawings is explained below.

Figure 5 illustrates one method of implementing the invention;

Figure 6 is a flow diagram representing a method for verifying a trusted path;

Figure 7 is a state transition diagram of a further embodiment utilizing various command states.

16

4. DESCRIPTION OF SPECIFIC EMBODIMENTS

A specific embodiment of the present invention is described with respect to a computer which supports various user processes and may operate in a multiuser environment. As an illustration, an implementation in connection with a VAX model computer, available from Digital Equipment Corporation (DEC), is briefly described. It should be understood that the present invention may be readily applied to various types of computer systems including PC networks, stand-alone multiuser computer systems, or single user computer systems, to name a few examples.

4.1   Overview of Software Architecture

FIG.1 illustrates a preferred computing environment which is generally designated by the reference numeral 2. A trusted computing base (TCB) 10 resides at the center of the computer system. The TCB 10 acts as a reference monitor in that access to information is mediated by the TCB 10.

In a general embodiment of the invention, the TCB 10 may be a secure kernel residing under the general operating system of a computer system. In one embodiment, the TCB 10 comprises trusted code residing underneath a VMS or Ultrix system.

17

The TCB 10 enforces a security policy which identifies permissible modes of access between users or computer processes (subjects) and files, memory segments or other objects. Access to stored information is mediated by the TCB 10, so that the TCB 10 may determine whether a particular user or untrusted subject can access particular information. In order for the TCB 10 to fulfill its protective function, it must take into account the three requirements discussed above: (1) completeness, (2) isolation, and (3) verifiability.

A more complete understanding of the TCB 10 may be obtained by reference to FIG.2. As is illustrated, the TCB 10 may comprise several distinct layers. Each layer of the TCB 10 comprises trusted code. This trusted status is indicated by the security boundary 11. In one embodiment, layers of particular interest are the layers to be referred to as: secure server (SSVR) 12, Command Conveyor (CC) 14, and VTerm (VT) 16.

The TCB 10 includes the trusted code necessary to ensure compliance with the Orange Book requirements. In alternate embodiments, the trusted code may ensure compliance with different systems of security requirements. The TCB 10 also includes the code which actually executes the requested trusted command, as discussed below.

1

2   Each untrusted subject is generally supported by an untrusted

3   process. This process may spend time in one of two "modes." The first

4   mode, in which the process usually runs, is known as a "user mode."

5   This mode is a restricted hardware state such as the VAX user mode

6   that is known in the art. This process may also spend time executing

7   trusted code in an all-powerful hardware state. This state is represented

8   by the VAX "kernel mode", which is known in the art. This state may

9   be triggered by a request by the untrusted subject of the security kernel.

10   Such a request may be made in a conventional manner and is commonly

11   called a "kernel call." This mode may also be triggered by asynchronous

12   events (e.g. the completion of an Input/Output request previously made

3   by the untrusted subject).

14

15   In one embodiment, the TCB 10 also has one process associated

16   with each terminal that communicates with the user over the trusted

17   path. This process is known in the art as an "execution thread." In this

18   embodiment, each execution thread shares its address space with other

19   TCB execution threads. An advantage of such sharing is economy of

20   machine resources. Full-blown processes, each with its own address

21   space, would require far more machine resources than if such address

22   space sharing is employed.

23

1   It is the CC 14 which provides the interprocess communication

2   for passing parsed commands between the processes that support the

3   untrusted subjects, and the processes within the TCB 10 that support the

4   trusted path.

5

6   The untrusted code may submit to the TCB 10 the parsed

7   representation of the trusted command by making a kernel call. The

8   kernel call causes the process supporting the untrusted subject to switch

9   to the all-powerful hardware state and to begin to execute trusted code

10   designed to handle kernel calls. This trusted code recognizes the nature

11   of the kernel call and calls the CC 14. The CC 14 then obtains and

12   copies the binary representation of the trusted command into protected

3   memory and puts it in a global data structure that supports the user

14   process associated with the terminal in question.

15

16   The VT 16 layer includes the trusted code which monitors all

17   terminal activity and checks for activation of the secure-attention key.

18   In one particular embodiment, the VT 16 layer also includes code which

19   tracks which physical terminal is associated with a particular user or

20   untrusted subject.

21

22   The specific manner of implementation of the above described

23   activities will be apparent to one skilled in the art having the benefit of

24   this disclosure.

1

2   Referring to FIGS 1 and 3, "around" the TCB 10 is a general

3   untrusted operating system (UOS) 20. The UOS 20 generally comprises

4   untrusted code and may include a traditional operating system such as

5   Ultrix or VMS. Alternate constructions are envisioned where the UOS

6   20 includes any untrusted code. It will be appreciated that the system

7   may include untrusted code that is separate from TCB 10 and UOS 20.

8

9   Most user activity will occur in the UOS 20. In one embodiment,

10   a specific UOS 20 may run in each untrusted subject.

11

12   4.2   User Terminals and the Secure-Attention Key

13

14   Associated with the computer system are one or more user

15   terminals 30. These terminals 30 may be conventional in construction,

16   in that they allow the user to exchange information with the computer

17   system. Either "intelligent" or "dumb" terminals may be used. Examples

18   of such dumb terminals are the well-known VT series 200 and 300

19   terminals. These terminals are available from Digital Equipment

20   Corporation or its authorized dealers.

21

22   A particularly useful terminal includes control programming that is

23   designed with security features in mind, such as is described in a co-

24   pending U.S. patent application entitled "Method for Securing Terminal

21

and Terminal Apparatus for Use with the Method," Serial No. 456,672,
filed by Wei-Ming Hu, Clifford E. Kahn, Andrew H. Mason, and Peter
A. Sichel on December 26, 1989 and to be commonly assigned with this
application.

A keyboard is attached to each user terminal. Each keyboard
includes a key employed as a secure-attention key. This key is utilized
to establish a trusted path between a user terminal and the TCB 10. In
one embodiment, the secure-attention key is an "out-of-band key" for the
UOS 20. It is desirable that the signal sent by the terminal upon
activation of the secure-attention key is generally not utilized by the
applications running in the UOS 20. An out-of-band key is convenient
since it does not restrict the traditional number of user keys and the
signal is sent immediately, even if other I/O is pending. For example,
the secure-attention key may be the BREAK key on the terminal
keyboard.

In many terminals (such as the VT 2XX and VT 3XX families)
locking of the keyboard prevents the transmission of several key signals.
In these terminals, however, the BREAK key signal may be transmitted,
even from a locked keyboard. Thus it is generally desirable to employ
the BREAK key as the secure-attention key. Also, the signal generated
by activation of the BREAK key comes in ahead of other characters in
the keyboard buffer. A further advantage of utilizing the BREAK key is

that software running in the UOS 20 cannot get the terminal to send that character.

4.3   The Trusted Path and Direct Commands

The TCB 10 generally operates as a reference monitor.

Upon initial login or at other times when the user is not connected to a process in the UOS 20 (e.g., an application program), the TCB 10 responds to a comparatively limited series of "direct commands," which may include a CONNECT command to connect to an untrusted subject. When the user is connected to such an untrusted subject, the TCB 10 checks for the secure-attention signal; if a signal other than the secure-attention signal is received, the TCB 10 passes the signal to the connected untrusted subject.

As discussed above, in one specific embodiment the trusted code in the VT layer 16 monitors all terminal activity. When the code in the VT 16 notices that the BREAK key has been depressed, the SSVR is signaled in a conventional manner (such as by advancing an event count or activating a flag). Code in the SSVR 12 recognizes this signaling and establishes a trusted path between the user terminal and itself. In this manner, the secure-attention key signal is trapped by the TCB 10.

1   In certain security environments, general security requirements

2   mandate that the TCB 10 make displays when a trusted path is

3   established and exited by the user. In one embodiment, when a trusted

4   path is established, the TCB 10 sends a reset command sequence to the

5   physical user terminal 30 to assure that it is in a known state. This is

6   done in part to ensure that the information displayed to the user by the

7   TCB 10 is in the intended form, (e.g., correct font). A particularly

8   useful sequence of commands to implement the reset command sequence

9   is described in the aforementioned copending U.S. patent application by

10   Hu et at.

11

12   When the secure-attention key is activated, other than in

3   connection with the issuance of a trusted command, the secure server 12

14   displays a particular banner and prompt (for example: "SSVR>" together

15   with the user's process identifier, discussed below), to indicate that the

16   user is conversing interactively with trusted code. If the secure-attention

17   key was activated in connection with the issuance of a trusted command,

18   the secure server 12 displays to the user some indication of what is to

19   be done, as discussed below.

20

21   When command execution is complete, or when the user

22   establishes or reactivates a session with the UOS 20, the TCB 10 issues

23   a message to inform the user that he/she is breaking the trusted path.

24   When a session with the TCB is complete, the memory in the physical

terminal may be cleared by the TCB. This is done to protect trusted

information from both untrusted code and unclassified persons who may

later use the terminal. A particularly advantageous method of clearing

the terminal memory is described in the aforementioned co-pending

patent application.

The number of direct user commands available from the TCB 10

is preferably limited to reduce the complexity of the TCB 10. Generally,

a command is direct if:

      a. one needs the command when one

is not connected to an untrusted subject;

      b. end users depend on the com-

mand's correct behavior for security (e.g.,

LOGOUT); or

      c. the command must be issued when

no user process in the UOS 20 is running.

In most cases, the user will converse with the TCB 10 through the

SSVR 12. In one embodiment the user conversing with the SSVR 12

may be restricted to commands which allow him/her to establish

(connect) or end (disconnect) a computing session in the general UOS

1   20. Further commands may be added, such as allowing the user to view

2   the existing UOS 20 sessions or change his or her password, but it is

3   generally desirable to limit the number of available commands in the

4   TCB 10. It is further desirable to ensure simplicity in the syntax of

5   these direct commands.

6

7   The simple syntax of the direct commands, and the comparatively

8   small amounts of code necessary to interpret these commands, do not

9   substantially increase the complexity of the TCB 10.

10

11 4.4   Login Processing by the TCB

12

13   To initiate a computing session in the illustrative UOS 20, the

14 user may first login to the computing environment through the SSVR 12.

15 To initiate a login, the user activates the secure-attention key (or

16 performs some other action to gain the attention of the SSVR 12). As

17 discussed above, this establishes a trusted path between the user terminal

18 and the TCB 10. Alternate embodiments are envisioned wherein other

19 methods are employed to establish a trusted path.

20

21   Once a trusted path has been established, the SSVR 12 prompts

22 the user for his/her specific user name and password. The various

23 layers in the TCB 10 perform well-known security checks to ensure that

24 the user has the necessary access rights to login at the terminal. If the

1   user possesses the required access rights, a computing session is

2   established in the SSVR 12.

3

4   At this time, the user's security information is stored by the TCB

5   10. The TCB 10 also stores in memory a unique identifier of the user

6   process. This information is maintained in memory by the TCB 10 as

7   long as the user has an active session.

8

9   4.5   Connection to Untrusted Process

10

11   Because the number of direct user commands in the TCB 10 is

12   preferably limited, the user will generally choose to establish a

3   computing session with an untrusted subject. Such an untrusted subject

14   may include application programs residing in the UOS 20.

15

16   In one specific embodiment, when the user establishes a

17   computing session in the UOS 20, the user's physical terminal 30 is

18   associated with a virtual terminal of the UOS. The user converses with

19   the untrusted subject through a computing session within the UOS 20.

20   When such a connection with an untrusted subject is made, the VT layer

21   16 of the TCB 10 stores in memory which physical terminal 30 is

22   associated with which virtual terminal in an untrusted subject. In this

23   manner, trusted commands received from an untrusted subject may be

24   attributed to the actual user at the physical terminal 30. Once the user

27

1   has established a computing session within the UOS 20, normal

2   computing activities may be carried out.

3

4   A more detailed understanding of these activities may be obtained

5   through reference to FIG.4A and 4B. Activities indicated by heavily

6   bordered blocks are performed by the TCB 10, or trusted code.

7   Activities represented by lightly bordered blocks are performed by

8   untrusted code.

9

10   In FIG.4A, the user terminal is represented as having established

11 a computing session 100 within the general UOS 20. During the session,

12 the user may attempt to execute various commands. Utilities within the

3   UOS 20 determine whether the issued commands are trusted commands

14 50. If the command is not identified by the UOS 20 as a trusted

15 command, the UOS 20 attempts to perform the command at process 60.

16 In these cases the operation and existence of the TCB 10 may have little

17 effect on the command's behavior as seen by the user.

18

19 4.6   The Process Identifier

20

21   In one embodiment, a process identifier (process ID) is associated

22 with each user process at the time of login. This process ID is a

23 pseudo-randomly generated alpha-numeric tag which is associated with

24 the user throughout his/her computing session.

1   The process ID is useful if the untrusted code can temporarily

2   disable or delay the effect of activation of the secure-attention key. For

3   example, untrusted code can send a reset command to some terminals.

4   While these terminals are resetting, activation of the secure-attention key

5   has no effect. Such a technique may allow untrusted code to trick the

6   user into thinking a trusted path had been established when it had not

7   been. The process ID inhibits such trickery by allowing the user to

8   distinguish between actual and emulated trusted paths.

9

10   When the user first logs in, he/she will not yet know the process

11   ID and cannot detect a false trusted path. However, to simulate a login

12   dialogue, untrusted code would have to guess when the user activated

3   the secure-attention key, or somehow detect the activation of the key

14   even though the TCB could not detect it. Guessing the activation time

15   is impracticable, because users start login dialogues at their own

16   convenience and give no advance signal to the TCB prior to activation

17   of the secure-attention key. Detecting the secure-attention key even

18   though the TCB could not detect it would be possible only by causing

19   the key to yield a signal different than the TCB expects. Methods to

20   prevent such an event are described in the co-pending application by Hu

21   et al., referenced above.

22

23   With some terminals, the untrusted code may be able to program

24   the terminal so that the secure-attention key generates a signal that is

1   recognizable to the untrusted code, but not to the TCB. In these cases,

2   generation of the process ID at the time of login would be

3   unsatisfactory. Instead, the process ID may be generated prior to user

4   login. For example, in these embodiments, the process ID may be

5   generated when the user is registered with the TCB 10 and stored in

6   protected memory. In this embodiment the user may be given a way of

7   changing his/her process ID, just as the user can generally change their

8   password.

9

10   In one particular embodiment, when the user logs in the SSVR 12

11   pseudo-randomly generates a process ID and assigns this ID to the

12   specific user at the physical terminal. Methods of generating pseudo-

3   random identifiers are generally known in the art. In one embodiment,

14   the random number initial seed is based on a hardware (as opposed to a

15   software) event. The algorithm for generating successive random

16   numbers should be computationally difficult to invert.

17

18   This process ID is maintained in the TCB and is never provided

19   to untrusted code. Thus, untrusted code in the general UOS 20 is

20   prevented from discovering the specific process ID assigned to the user.

21

22   The user is responsible for noting and retaining his/her assigned

23   process ID. To promote user retention the process ID may be

24   generated as a pronounceable group of characters. Methods for such

generating pronounceable passwords are described in "A Random Word

Generator", MITRE Technical Report (MTR-3006), May, 1975, by

Morrie Gasser.

The process ID is displayed to the user when a trusted path is

established between the user terminal and the SSVR 12. Displaying the

ID serves a useful security function as it inhibits untrusted code from

"spoofing" the user. Absent the process ID it may be possible for an

untrusted subject to emulate the SSVR 12. An untrusted subject,

emulating the SSVR 12, may be able to trick the user into divulging

security-relevant information or otherwise compromising system security.

The process ID, known only to the user and the SSVR 12,

inhibits this type of "spoofing." Before the user responds to an apparent

SSVR 12 request, he/she will first verify that the proper process ID is

displayed. Since the process ID is known by only the user and the

SSVR 12, it is extremely difficult for an untrusted subject, ignorant of

the process ID, to fool the user into believing that he/she is conversing

with the SSVR 12.

FIG.6 illustrates a flow diagram of this process. When the user

initially logs into the computer system through the SSVR 12, a randomly

generated process ID is assigned to the user. This activity is represented

by process 310. In one embodiment, this process ID comprises

31

1   alphabetic characters and is pronounceable. The pronounceable nature

2   promotes user retention of the ID. The process ID is then stored in

3   trusted memory by the SSVR 12 and displayed to the user by the SSVR

4   12 at process 320.

5

6   During the course of the computing session, the TCB determines

7   if the user is operating through a trusted path. This activity occurs at

8   process 330. If the TCB determines that a trusted path is not

9   established, the process ID will not be displayed to the user. This is

10   noted at process 350.

11

12   The process ID is maintained in trusted code and memory, and is

3   inaccessible to untrusted subjects. Each time a trusted path is

14   established between the SSVR 12 and the user, the SSVR 12 displays

15   the user's process ID at process 340. By observing the displayed ID, the

16   user is assured that an actual trusted path has been established.

17

18   4.7   The Command Conveyor

19   As discussed above, the command conveyor (CC) 14 is a layer in

20   the TCB 10 which comprises part of the trusted code. The CC 14 is

21   utilized to ensure proper communication of trusted commands between

22   the SSVR 12 and the process supporting the untrusted subject. As

23   discussed, such an untrusted subject may include a UOS 20.

24

1   Associated with each physical terminal 30 is a command buffer.

2   In some embodiments, a status code may be associated with each

3   terminal. In a still other embodiments, one of five command states is

4   also associated with each physical terminal. The command buffer is

5   used to store the binary representation of the trusted command. The

6   status code may be employed to provide the resulting execution status to

7   the untrusted subject. The command states generally comprise five

8   possibilities:

9

10   (a) no command,

11   (b) command submitted,

12   (c) command in progress,

3   (d) ignore finish, and

14   (e) command done.

15

16   These states are used to indicate the various states that may be

17   encountered in the execution of a trusted command. FIG. 7 is state

18   transition of the command states. A detailed explanation of these states

19   and the statuses may be obtained from the following discussion.

20

21   4.8   Distribution of Processing of Trusted Commands

22

23   As discussed above, the TCB 10 "recognizes" two categories of

24   commands. For the purposes of this specification "commands" include

1   user instructions received through the use of command line interfaces,

2   menu interfaces, direct manipulation interfaces or the like.

3

4   The first category includes "direct commands" issued while the

5   user is operating in the SSVR 12 layer of the TCB 10, i.e., is

6   interactively conversing with the TCB 10. As discussed above, it is

7   desirable to limit the number of direct commands available to the user.

8   The execution and processing of these direct commands is carried out

9   using traditional methods known in the art.

10

11   The second category includes "trusted commands." These

12   commands are issued by a user operating in a non-trusted computing

3   environment, such as in the general UOS 20. In one embodiment these

14   trusted commands are issued by a user operating in an untrusted virtual

15   machine. FIGS.4A and 4B illustrate the activity necessary to execute

16   such a command.

17

18   If the user attempts to execute a trusted command, the parser

19   residing in the general UOS 20 parses the trusted command string and

20   converts it into a binary representation at process 70. In parsing the

21   command, the untrusted code in the UOS 20 checks syntax and

22   semantics, and prompts for missing parameters. Parsing strategies are

23   well known in the art and will not be discussed herein.

24

1   From a security perspective, it is not required that the code in the

2   UOS 20 perform an accurate translation of the user's commands into a

3   parsed representation (binary representation). The confirmation

4   mechanism, described below, is designed to catch such errors whether

5   benign or malevolent.

6

7   Following parsing of the command, a process supporting the UOS

8   20 issues a call to the TCB, submitting to the TCB the identification of

9   the terminal from which such command was issued, at process 80. In

10   one embodiment, the virtual terminal from which the command was

11   issued is submitted to the TCB. Such a call is processed by the CC 14

12   in the TCB 10. The CC 14 then determines which trusted subject

3   submitted the command and copies the parsed representation into

14   protected memory at process 90. In one embodiment this determination

15   is made by asking VTerm to identify the physical terminal associated

16   with the UOS's virtual terminal.

17

18   Such a submission of the binary representation will be successful

19   only if the TCB 10 determines that the submitting terminal is connected

20   to an active process.

21

22   Normally, in embodiments utilizing the command states, the

23   command state of the submitting physical terminal is no command, which

24   indicates that there is no trusted command submitted by that physical

1   terminal or in progress. This state is represented by state 400 in FIG. 7.

2   When the state is no command, the CC 14 saves the binary

3   representation in the command buffer, and sets the physical terminal's

4   state to command submitted, represented by transition state 410.

5   Otherwise, the submission fails.

6

7   When the untrusted subject running in untrusted code receives

8   indication from the TCB 10 that the command has been submitted, it

9   requests the user to activate the secure-attention key. In one

10 embodiment, the untrusted code in the UOS 20 will request that the

11 user activate the BREAK key.

12

3   If the user activates a cancellation key, for example Ctrl-Y or

14 Ctrl-C, instead of activating the secure-attention key, code in the general

15 UOS 20 may cancel the trusted command. This can be accomplished by

16 the untrusted code issuing a kernel call (i.e., a request by the untrusted

17 subject of the security kernel) and informing the CC 14 that the

18 command is to be aborted. If the CC 14 finds that the command has

19 not been sent to the SSVR 12, the command is canceled. Similarly, if

20 the user process in the UOS 20 is killed, the process rundown code may

21 cancel the trusted command request. However, if the CC 14 finds that

22 the command has already been sent it proceeds; because the user is

23 already executing the command.

24

1   In embodiments employing command states, cancellation of the

2   trusted command request results in the command state being changed

3   from command submitted to no command. This process is indicated in

4   FIG. 7.

5

6   If the user attempts to establish a trusted path at the same time

7   the trusted command is sent to the TCB 10, then the TCB 10 must

8   arbitrate; either of these requests may be processed first. If the secure-

9   attention signal is processed first, the TCB 10 may defer the command

10   submission, and a connection with the SSVR 12 is established. If the

11   submission is processed first, then the user sees the confirmation screen

12   as is described below.

3

14   If, when prompted, the user activates the secure-attention key, a

15   trusted path is established between the user's physical terminal 30 and

16   the SSVR 12 in the TCB 10. This configuration is illustrated by the

17   dark line in FIG.3. When the VT layer 16 detects the secure-attention

18   (BREAK) key, it notifies the SSVR 12 which then establishes a trusted

19   path in the trusted path-supporting process associated with the user's

20   physical terminal 30.

21

22   Once the trusted path is established, the TCB 10 maintains

23   exclusive control over the physical terminal 30. In one embodiment,

24   untrusted code in the UOS 20 waits to be notified that the command

1   has been completed. Thus, the user essentially bypasses the untrusted

2   code, and converses only with trusted code. Thus, if the user's process

3   in the general UOS 20 is killed, or the user's process terminates, or the

4   general UOS 20 crashes, the trusted command may still be executed.

5   The user will not be informed of any of these activities until the trusted

6   command has been completed or abandoned.

7

8   Following establishment of the trusted path, the security

9   requirements of the system require interaction between the user and the

10   SSVR 12. This is necessary to execute commands which require the

11   establishment of a trusted path. The Orange Book criteria, and/or the

12   system's security goals, will determine which commands require this

·3   interaction as is generally discussed in Section 4.9 below.

14

15   The establishment of a trusted path in these instances is useful

16   for several reasons. First, it allows the secure server to identify the

17   specific user issuing the command. Second, the trusted path allows the

18   user to review and approve (or reject) the requested action in a trusted

19   fashion. This reduces the possibility of malicious untrusted software

20   altering a command request, and allows for the detection of parsing

21   errors.

22

23   FIG.4B illustrates the activity occurring after control has passed to

24   the TCB 10. As discussed above, the TCB 10 first establishes a trusted

1   path at process 120. The TCB then checks to see if a trusted command

2   has been submitted by the user. Untrusted code may then wait for the

3   command to finish.

4

5   Once a trusted path has been established, the SSVR 12 will call

6   the CC 14 to return the binary representation at process 130. In one

7   embodiment, the CC 14 will first determine the command state of the

8   physical user terminal 30. In a still further embodiment if the command

9   state is command submitted 410 (FIG. 7), the CC 14 will return to the

10 SSVR 12 the binary representation, and change the command state to

11 command in progress 420. If the state is no command 400 or command

12 done 430, the call will fail, and the SSVR 12 will display the SSVR

3   prompt (SSVR>) to the user. An initial command state of command in

14 progress 420 or ignore finish 440 implies a bug in the SSVR 12 and may

15 result in the CC 14 crashing the TCB 10.

16

17   In these embodiments, if the command state was command

18 submitted 410, then the CC 14 will return to the SSVR 12 the binary

19 representation of the requested command.

20

21 4.9   Display of Command to User

22

23   Once the parsed command has been returned to the SSVR 12,

24 the SSVR 12 will determine if the command requires confirmation at

39

1   process 140 (FIG. 4B). Confirmation should generally be required for

2   commands which request modification of information. Commands that

3   do not alter the information, such as the viewing of information, should

4   not generally require confirmation. For purposes of confirmation,

5   redirecting the display into a file should be considered as viewing

6   information.

7

8   The decision of whether a given command should require

9   activation of the secure-attention key to execute depends on the basic

10   security policy of the system. In one embodiment, secure-attention

11   trusted commands are a mechanism for use by system managers, security

12   managers, operators, and holders of special privileges.

3

14   Generally, a command which is not a direct command, is a

15   secure-attention command if:

16

17   a. it modifies a security kernel

18   database in a security-relevant way; or

19

20   b. it displays information on which a

21   systems manager, security manager, or

22   operator will base a security decision.

23

1   If the command is one that does not require confirmation, the

2   TCB 10 enforces the system's security policy and determine if the user

3   has the necessary access rights to execute the command at process 145.

4   If this is the case the TCB 10 attempts to execute the binary

5   representation at process 150. Since certain commands, e.g. commands

6   that merely display information, do not change the state of the system,

7   user confirmation is not required. There is no security problem in not

8   confirming these commands since the system only allows users to view

9   information to which they have access.

10

11   There are certain commands that change the state of the system

12   may not require confirmation. Some commands may allow the user to

3   turn on a file to receive the user's redirected output. This files are

14   sometimes known in the art as "listing files." Generally, the turning on

15   of a listing file is a confirmed command.

16

17   In situations where a listing file has been turned on (i.e., the

18   output is redirected into this file) commands which normally display

19   information may, by writing to the listing file, alter the state of the

20   system. In these cases, confirmation of these commands is generally not

21   required because the flow of information is constrained by the TCB's

22   security rules.

23

When a command requesting the display of information is
executed by the TCB 10, the SSVR 12 responds by displaying a
complete representation of the requested command as well as the
requested information, i.e. the standard human-readable representation of
the command. In one embodiment, the SSVR 12 displays the function
code, command modifiers, and command parameters all in unambiguous
notation. If employed, the process ID may also be displayed.

Thus, if the user's command is improperly parsed, or modified in
an unauthorized manner, the user will be able to observe the difference.
This inhibits an untrusted subject from fooling a user into believing that
he or she is observing the results of a submitted command, when in fact
that command had actually been altered.

If the command requested is one for which confirmation is
required, the SSVR 12 displays a standard human-readable
representation of what is about to be done to the user terminal at
process 170. The code representing each command should produce a
different display, unless the commands have the same effect. As
previously discussed, the re-display of the action that a binary
representation will perform is less complex than the parsing of the
original command. This is in part because by definition there is only
one standard human-readable representation for each parsed command.

By parsing the command in untrusted code, and verifying the request in trusted code, the overall complexity of the TCB 10 is decreased. The display of "what is about to be done" may of course vary from one command to another.

If the command's purpose is to add a record to a database or create an object, the display may contain what the new record will contain, or what the attributes of the object will be if the command is confirmed.

If a particular database or object is to be modified, the display may show what the updated data will look like, or what the attributes of the object will be if the command is confirmed. For commands which remove an existing object, or delete an existing file, the display may show the record or object.

Generally, for both adding/creating and modifying, the data packet representing the submitted command may contain fields for the objects attributes, such as its name, its size and its protection. For each of these fields, the packet may also have a flag indicating whether the field is meaningful. These flags indicating meaningful fields may be know as "field selectors."

In add/create commands, the TCB may supply default values for ⸗

unselected fields. In commands which modify a database entry or object,

unselected fields may retain their previous values.

To aid the user, the selected fields may be highlighted when

displayed for confirmation. Such highlighting may be accomplished by

methods known in the art (e.g., by using escape sequences that begin

and end reverse video). This highlighting draws the users attention to

the fields that have non-default or changed values and helps to protect

the user from tricks by untrusted code.

When an audit record is produced of the confirmation display via

the sink (discussed below) the reverse-video escape sequences may be

replaced with dashes on the next line. Example:

Name:JOHN.DOE
- - - -

This change permits the audit record to be properly displayed on

a wide variety of devices, even simple line printers.

Alternative embodiments are envisioned in which the TCB checks

each field's value: for add/create commands the fields that have non-

default values are highlighted; and for modify commands the fields that

1 are changing are highlighted. This "field comparison" approach generally

2 results in less highlighting.

3

4   Generally, for every command, every packet field is displayed on

5 the screen, except for fields ignored by the particular command.

6

7   In the cases when a database will not be modified, the display

8 generated by the SSVR 12 may express what action is to be done. In

9 this manner, the possibility of an erroneous confirmation is reduced since

10 the user is requested to confirm the results, not the command.

11

12   The SSVR 12 may also display a list of user protection attributes

.3 so that the user knows what role he or she is exercising. The user is

14 required to confirm explicitly (Yes or No) the action requested. In a

15 general embodiment, the TCB 12 displays for confirmation the results

16 which would occur if the command is executed.

17

18   Generally, any information in the binary representation that is

19 displayed is first validated. This information is checked once to ensure

20 that certain requirements are met, e.g., to ensure that the counts of

21 array elements are within array bounds (since out-of-bounds conditions

22 could provide an opportunity for entry into the TCB by illicit software).

23

1   Before displaying the contents of a packet containing the binary

2   representation of the command, anything that could cause the formatting

3   code to malfunction may be validated. Generally, such candidates for

4   validation could include anything used to index into something else.

5   Specific examples include the PASCAL language's implementation of

6   varying strings (a count followed by an array of characters; the count is

7   checked to be less than or equal to the character array size), and

8   numbers that represent keywords (their value is checked before it is

9   used to index into an array of keyword text strings). Further methods of

10   validation are well known in the art and will not be discussed herein.

11

12   In one embodiment, the replies that the user may make to the

3   confirmation prompt are "yes", "no", Ctrl-C, Ctrl-Y, or the BREAK key.

14   No default responses exist in this embodiment. For each user response,

15   the secure server may display a message restating the response. If the

16   response was "yes" or "no", the secure server 12 displays an indication of

17   the computing session in the UOS 20 when it is being be resumed after

18   command completion.

19

20   If, however, the user activates the Ctrl-Y, Ctrl-C, or the BREAK

21   key, the command will be aborted. In one embodiment, the CC then

22   changes the command status code to negative. This may be done to

23   inform the submitting process that the command has been aborted. In

24   these cases a direct session with the SSVR 12 will be established at

process 300. In these instances, the user must explicitly issue a resume command to return to the session in the UOS 20.

If the user confirms the results or command, by replying with "yes", or if the command does not require confirmation, the TCB 10 enforces the system's security policy at process 145. At this process the TCB 10 determines if the user has the necessary access privileges to perform the requested activity. If the user has the required access rights, the SSVR 12, along with other code in the TCB 10 attempts to execute the command in a conventional manner.

Upon attempted execution of the command, the SSVR 12 may inform the CC 14 of the execution status. In some high-security situations, concern for information flow from trusted to untrusted code will prevent informing the CC 14. The CC 14 may change the command status to indicate the SSVR 12 response. In addition, in alternate embodiments, the CC 14 changes the command state from command in progress 420 (FIG. 7) to command done 430. In these cases, the CC 14 notifies the submitting untrusted subject by traditional methods (e.g., by advancing an eventcount for that process). The SSVR 12 then reestablishes the terminal connection with the computing session in the UOS 20 from which the command was executed.

1   In embodiments utilizing the status code, if the user responds with

2   "no", the secure server informs the session in the UOS 20 that the

3   command was not executed (negative status code). If command states

4   are also employed, the SSVR 12 changes the command state to

5   command done, and re-establishes the user's session in the general UOS

6   20 at process 160.

7

8   If the trusted command requested by the user is a command

9   which affects one of the system databases, and the database record was

10   changed by some other process or user between the time the

11   confirmation display appeared and the current user confirmed the

12   command, then the SSVR 12 displays a message indicating that a change

3   has been made and requests reconfirmation. In this case, the SSVR 12

14   waits for a response as if it were the first time the information was

15   displayed.

16

17   To ensure that if two users attempt to modify the same element

18   at the same time, only one request is rejected, the TCB may utilize

19   identifiers known as "modification counts" (mod-counts).

20

21   Each element of key databases (or registries) may contain a

22   modification count. This mod-count is advanced after each modification

23   and thus identifies how many modifications have been made to the

24   database element. Thus, when a user attempts to modify a database

element, the SSVR 12 first obtains that database element's mod-count. ⸺ ⸺

After the user has confirmed the request, the TCB 10 checks the mod-count in the request against the current mod-count.

A different mod-count indicates that a change has been made to the database element, and the user is asked to reconfirm the request since the result previously confirmed by the user is not what the result would actually be.

To deter malicious users or subjects from preventing the execution of trusted commands, two measures may be taken. First, certain access rights may be required to change certain database elements. Second, the mod-count should not be advanced when a modification is requested that does not change the database element. In this manner, attempts by malicious subjects to prevent execution of a trusted command, by rapidly and spuriously causing the mod-count to be incremented, will be inhibited.

Once activity in the SSVR 12 is completed, the user's session in the general UOS 20 is reestablished. In one embodiment, this may be accompanied by the CC 14 informing the untrusted subject of the status of the submitted command. The untrusted code in the general UOS 20 then wakes-up the untrusted process running in the UOS 20 at process 160. In one embodiment, before the TCB 10 passes the user to the

1   untrusted subject, the SSVR 12 prompts the user to activate a carriage

2   return. This step is necessary to inform the user that he/she is being

3   placed into an untrusted environment.

4

5   4.10   <u>The SSVR as an Audit Recorder</u>

6

7   In one embodiment, the SSVR 12 audits (i.e., creates an audit

8   trail by writing to an audit or log file) at the command level. For direct

9   commands the SSVR 12 audits both the command line and the final

10   status of the command. For trusted commands (secure-attention

11   commands), the SSVR 12 audits both the confirmation display and the

12   final status of the command.

13

14   To record the confirmation displays, the SSVR 12 defines a

15   portion of memory to store the strings of information sent by the SSVR

16   12 to the user's terminal. In one embodiment, this portion of memory is

17   known as the "sink." Only the strings that comprise the SSVR's

18   confirmation screens are placed in the sink.

19

20   Strings may be appended to the sink only when the sink is

21   opened. The sink is opened when the SSVR 12 begins to service a

22   trusted command. The sink may be closed after the user has responded

23   to the confirmation prompt, or before the display of certain requested

information. The information stored in the sink does not include the user input to any prompts.

The information stored in the sink is maintained for auditing purposes and may be utilized according to traditional auditing practices.

4.11 Confirmation by Others than the Submitting User

In some situations it may be desirable for users other than the submitting user to confirm a secure-attention command. Such a situation may arise if there is a certain application or command file that includes a privileged command but is needed to be run by a large number of users. In the cases where it may be impracticable to grant all of the users the necessary access rights, it may be possible to grant one user the necessary rights, and to allow him/her to confirm this command for the other users.

Such non-submitting user confirmation may be accomplished by having the code in the untrusted subject lie about which user is submitting the command. For example, the code in the UOS 20 may ensure that certain commands are always sent to the TCB 10 as if they had been submitted by a selected user.

1   One purpose of this select user would be to confirm these certain

2   commands for the other users. After a non-select user submits one of

3   the certain commands, the selected user will be notified that there is a

4   submitted command waiting. The select user may then observe the

5   confirmation display. If the select user determines that the command

6   submitted was a legitimate command, he/she may confirm the command.

7   Otherwise, the original users attempt to execute the command fails.

8

9   4.12   Implementation Approaches

10

11   The method for executing trusted commands described above may

12   be implemented through the use of appropriate software. The actual

3   design and coding is a matter of routine for those skilled in the art

14   having the benefit of this disclosure. Such a design will of course vary

15   with the hardware platform and other factors associated with the specific

16   desired implementation.

17

18   FIG.5 illustrates one such method for implementing the present

19   invention. A programmable machine comprises a central processing unit

20   (CPU) 210, and an input/output (I/O) device 220. The I/O unit 220 is

21   capable of reading and transmitting stored programming instructions

22   from known storage media 230. The CPU 210 is capable of processing

23   these instructions and executing the programming steps represented on

1   media 230. Such systems are well known in the art and will not further

2   be described.

3

4   To implement the invention, the necessary programming steps are

5   stored on storage media 230 by traditional methods. The storage media

6   230 is then read by the I/O unit 220 and the programming instructions

7   are transmitted to the CPU 210. The CPU 210 reads and interprets the

8   instructions, and may thereafter carry out the method of the invention.

1 <u>CLAIMS</u>

2

3 WHAT IS CLAIMED IS:

4

5 1. A machine-executable method for executing a trusted command

6 issued by a user, said method comprising the steps of:

7

8 (a) parsing the trusted command in an untrusted computing

9 environment to generate a parsed command;

10

11 (b) submitting the parsed command to a trusted computing

12 environment; and

13

14 (c) executing the parsed command in the trusted computing

15 environment.

16

17

18

19 2. A method including the steps of claim 1 and additionally including

20 the steps, executed after step (b) of claim 1, of:

21

22 (1) in the trusted environment, displaying a representation of

23 the parsed command to the user;

24

1   (2)   receiving a signal from the user signifying whether the

2   displayed representation accurately represents the user's

3   intentions;

4

5   (3)   if the signal signifies that the displayed representation does

6   not accurately represent the user's intentions, then

7   preventing the performance of step (c) of claim 1.

8

9

10

11   3.   The method of claim 2 wherein the representation of the parsed

12   command is displayed, and the signal from the user is received,

'3   through a trusted path.

14

15

16

17   4.   The method of claim 1 wherein the trusted computing

18   environment comprises a security kernel.

19

20

21

22   5.   The method of claim 1 wherein the untrusted computing

23   environment comprises a general operating system.

24

1

2

3   6.   A method for executing in a computing system a trusted command

4   issued by a user, said method comprising the steps of:

5

6   (a)   receiving user identification data from the user via a

7   trusted path;

8

9   (b)   receiving the trusted command from the user via an

10   untrusted path;

11

12   (c)   parsing the trusted command in an untrusted computing

3   environment to generate a parsed command;

14

15   (d)   submitting the parsed command to a trusted computing

16   environment;

17

18   (e)   in the trusted computing environment, performing a security

19   check on the parsed command and user identification data;

20   and

21

22   (f)   in the trusted computing environment, executing the trusted

23   command.

24

7. The method of claim 6, wherein the security check enforces an Orange Book security criterion.

8. A method including the steps of claim 6 and additionally including the steps, executed after step (d) and before step (f) of claim 6, of:

    (1) in the trusted environment, displaying a representation of the parsed command to the user;

    (2) receiving a signal from the user signifying whether the displayed representation accurately represents the trusted command; and

    (3) if the signal signifies that the displayed representation does not accurately represent the trusted command, then preventing the performance of step (f) of claim 6.

1

2   9.   A method including the steps of claim 6 and additionally including

3   the steps, executed after step (d) and before step (f) of claim 6,

4   of:

5

6   (1)   in the trusted environment, displaying a

7   representation of the parsed command to a second

8   user;

9

10   (2)   receiving a signal from the second user signifying

11   whether the displayed representation accurately

12   represents a legitimate command; and

3

14   (3)   if the signal signifies that the displayed

15   representation does not accurately represent a

16   legitimate command, then preventing the

17   performance of step (f) of claim 6.

18

19

20

21   10.   A method for ensuring the existence of a trusted path in a

22   computing system comprising the steps of:

23

1   (a)   in a trusted computing environment, upon login by a user,

2   assigning a process identifier to the user in the trusted

3   computing environment;

4

5   (b)   storing the assigned process identifier in trusted memory;

6

7   (c)   establishing a trusted path;

8

9   (d)   in the trusted path, displaying the process identifier to the

10   user; and

11

12   (e)   upon a subsequent entry into the trusted path, displaying

3   the process identifier to the user.

14

15

16

17   11.   The method of claim 10 wherein the process identifier is a

18   randomly or pseudo-randomly generated group of alphanumeric

19   characters.

20

21

22

23   12.   The method of claim 11 wherein the process identifier is

24   pronounceable.

1

2

3

4   13.   An automatic data processing machine programmed to execute the

5   method of any one of claims 1 to 12.

6

7

8

9   14.   An automatic data processing machine comprising means for

10   performing the method steps of any one of claims 1 to 12.

11

12

3

14   15.   A program storage device readable by a machine and tangibly

15   embodying a representation of a program of instructions adaptable

16   to be executed by said machine to perform the method of any

17   one of claims 1 to 12.

18

19

20

1   16.   Apparatus for executing a trusted command that is issued by a

2   user and that is parsed by untrusted parsing means to generate a

3   parsed command, comprising:

4

5   (a)   trusted means for receiving the parsed command; and

6

7   (b)   trusted means for executing the parsed command.

8

9

10

11   17.   Apparatus for controlling the execution by a machine of a trusted

12 ·   command that is issued by a user and that is parsed by untrusted

3   parsing means to generate a parsed command, comprising:

14

15   (a)   trusted-program storage means, readable by the machine,

16   for causing the machine to receive the parsed command

17   from the untrusted parsing means; and

18

19   (b)   trusted-program storage means, readable by the machine,

20   for causing the machine to execute the parsed command.

21

22

23

1	18. Apparatus for controlling the execution by a machine of a trusted command that is issued by a user with user identification data and that is parsed by untrusted parsing means to generate a parsed command, comprising:
2	
3	
4	
5	
6	(a) trusted program storage means, readable by the machine, for causing the machine to receive the user identification data from the user;
7	
8	
9	
10	(b) trusted program storage means, readable by the machine, for causing the machine to receive the parsed command from the untrusted parsing means;
11	
12	
3	
14	(c) trusted program storage means, readable by the machine, for causing the machine to perform a security check on the parsed command and a security check on the user identification data; and
15	
16	
17	
18	
19	(d) trusted program storage means, readable by the machine, for causing the machine to execute the trusted command.
20	
21	
22	
23	

19. Apparatus as in claim 18 and additionally comprising:

    (1)   trusted program storage means, readable by the machine, for causing the machine to display a representation of the parsed command to the user;

    (2)   trusted program storage means, readable by the machine, for causing the machine to receive a signal from the user signifying whether the displayed representation accurately represents the trusted command; and

    (3)   trusted program storage means, readable by the machine, for preventing the machine from executing the trusted command if the signal signifies that the parsed command does not accurately represent the trusted command.

20. Apparatus as in claim 18 and additionally comprising:

    (1)   trusted program storage means, readable by the machine, for causing the machine to display a representation of the parsed command to a second user;

    (2)   trusted program storage means, readable by the machine, for causing the machine to receive a signal from the second

1   user signifying whether the displayed representation

2   accurately represents a legitimate command; and

3

4   (3) trusted program storage means, readable by the machine,

5   for preventing the machine from executing the trusted

6   command if the signal signifies that the parsed command

7   does not accurately represent a legitimate command.

ABSTRACT

A method for executing trusted commands, in which a trusted command is first received from a user at a user terminal and parsed by untrusted code; then passed to a trusted computing base for execution. The trusted computing base displays some indication of what is to be done back to the user for confirmation. Confirmation of the commands prevents unauthorized modification of the commands and increases system confidence. A randomly (or pseudo-randomly) generated process identifier is employed to verify the existence of a trusted path.

g:\digu\012\08.toe

Figure 1

Figure 3

10

11

12 SSVR (Secure Server)

14 CC (Command Conveyer)

16 VT (VTerm)

Figure 2

```
100
```

Terminal
(UOS)

Trusted
Command? —— 50

No —— Perform
Command —— 60

Yes

Parse
Command —— 70

Submit Binary
Representation to TCB
(Kernel Call) —— 80

CC Copies Parsed
Command to
Memory Buffer
Associated with
User Terminal —— 90

Secure
Attention
Key? —— 110

Ctrl-C or
Ctrl-Y

Yes

```
200
```

Figure 4A

```
                    ┌─────────┐
                    │   200   │
                    └────┬────┘
                         │
            ┌────────────┴────────────┐        120
            │   Establish Trusted     │──────/
            │         Path            │
            └────────────┬────────────┘
                         │
                    ╱────┴────╲              130
                  ╱   Call for  ╲─────────/                    ┌──────────────┐  300
                 ╱    Parsed     ╲                             │     SSVR     │─/
                 ╲   Command    ╱   No Command or  ─────────▶  │    Prompt    │
                  ╲           ╱     Command Done                └──────────────┘
                    ╲───┬───╱                                         ▲
                        │                                             │
                  Command                                             │
                  Submitted                                           │
                    ╱───┴───╲           140                           │
                  ╱           ╲────────/                              │
          No     ╱ Confirmation ╲   Yes          170                 │
        ┌───────╱   required ?    ╲───────┐     /                     │
        │       ╲               ╱         │    / ┌──────────────────┐ │
        │        ╲             ╱          └──────│ Display Command to│ │
        │         ╲──────────╱                   │   User Terminal   │ │
┌───────┴──────────┐                             └─────────┬────────┘ │
│ Display Command to│                                      │          │
│   User Terminal   │                                 ╱────┴────╲ 180 │
└───────┬──────────┘                               ╱             ╲──/  │
        │                               Yes      ╱                ╲    │
        │◀──────────────────────────────────────╱   Confirmed      ╲───┘
        │                                        ╲        ?        ╱ Ctrl-C, Ctrl-Y,
        │                                         ╲              ╱   or BREAK
┌───────┴──────────┐   145                         ╲──────┬────╱
│ Enforce Security  │─/                                   │
│      Policy       │                                     │ No
└───────┬──────────┘                                      │
        │                                                 │
┌───────┴──────────┐   150                                │
│  Perform Command  │─/                                    │
└───────┬──────────┘                                       │
        │◀─────────────────────────────────────────────────┘
        │
┌───────┴──────────┐   160
│   Wake Up UOS     │─/
│   User Process    │
└───────┬──────────┘
        │
   ┌────┴────┐
   │   100   │
   └─────────┘
```

Figure 4B

CPU

I/O Device —220

—230

210

Figure 5

```
                    ┌─────────────────────┐
                    │       LOGIN         │ ──── 310
                    │                     │
                    │  - Process ID       │
                    │    Assigned to User │
                    └─────────────────────┘
                              │
                    ┌─────────────────────┐
                    │  - Store Assigned   │ ──── 320
                    │    ID in Secure     │
                    │    Memory           │
                    │  - Display Process  │
                    │    ID to User       │
                    └─────────────────────┘
                              ┆
                            ╱╲
                          ╱    ╲ ──── 330
                  No    ╱ Trusted ╲   Yes
              ┌───────╱   Path ?    ╲───────┐
              │       ╲            ╱        │
              │         ╲        ╱          │
              │           ╲    ╱            │
              │             ╲╱              │
              │                             │
    ┌─────────────────┐           ┌─────────────────┐
    │   Do Not        │ ── 350    │    Display      │ ── 340
    │   Display       │           │   Process  ID   │
    │   Process  ID   │           │                 │
    └─────────────────┘           └─────────────────┘
              │                             │
              │    ┌─────────────────┐      │
              └────│    Process      │──────┘
                   │   User Input    │
                   └─────────────────┘
                            │
```

Figure 6

NO
COMMAND
400

COMMAND
SUBMITTED
410

COMMAND
DONE
430

COMMAND
IN
PROGRESS
420

IGNORE
FINISH
440

Notation:

································ Unusual Path
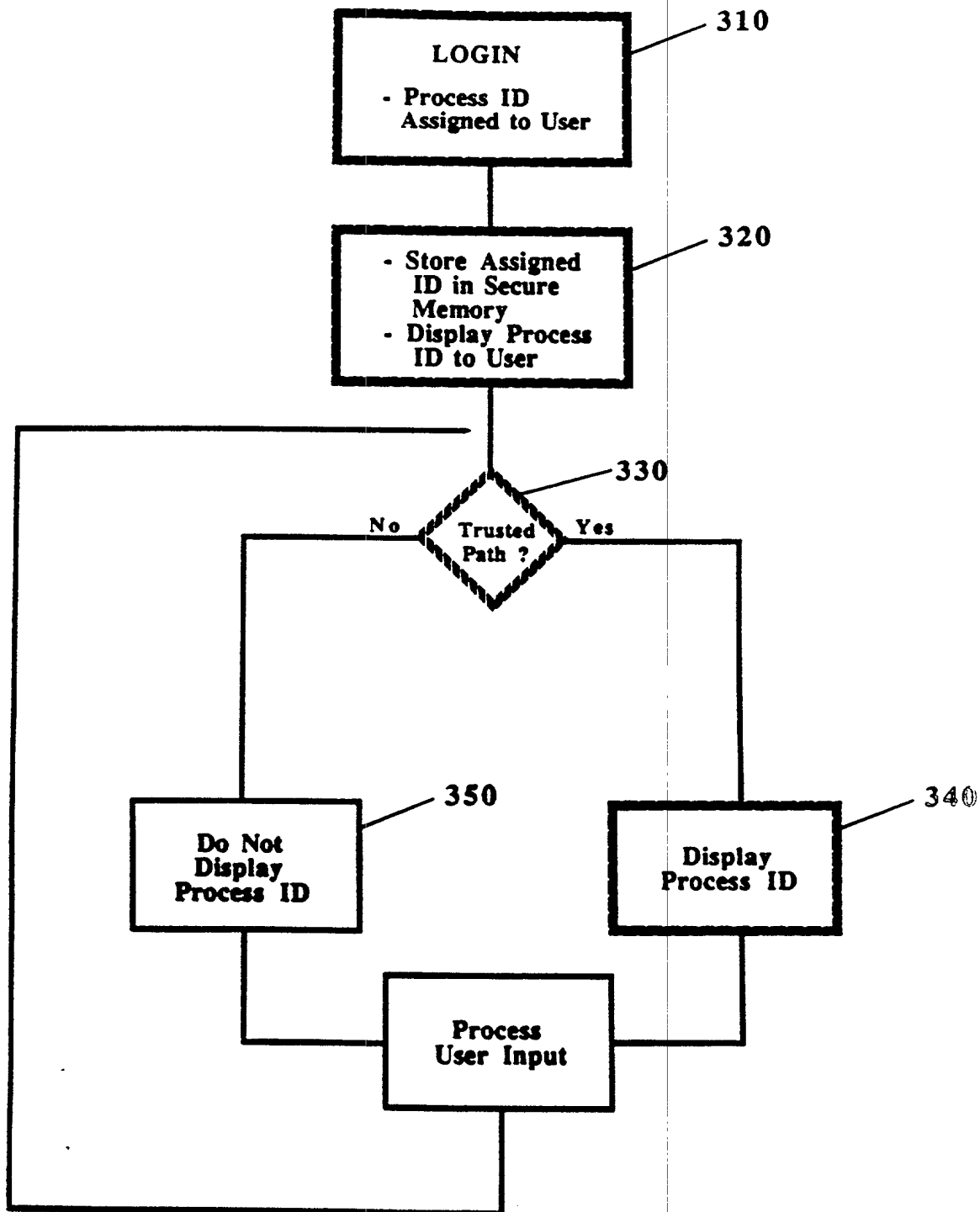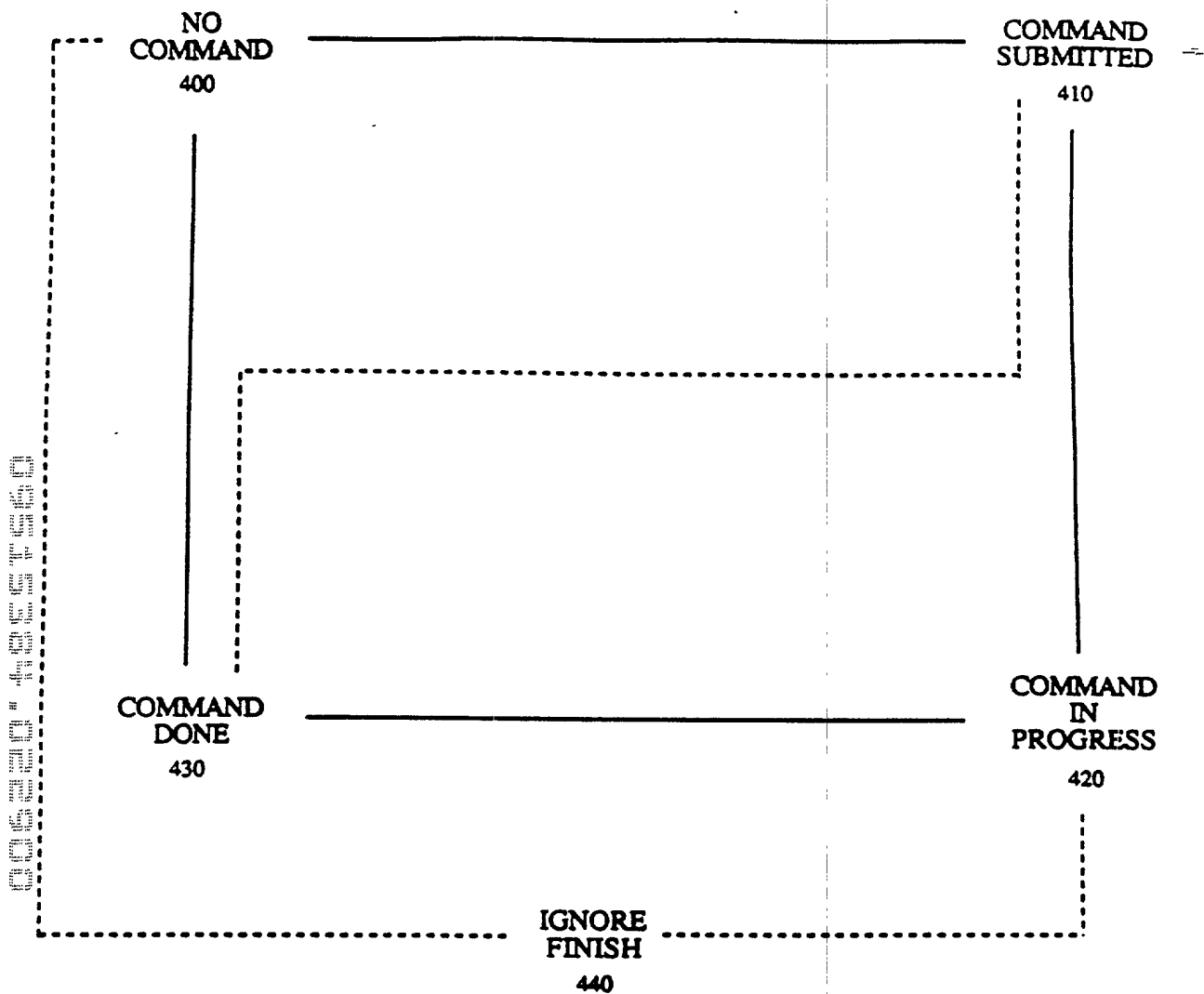
─────────── Normal Path

FIGURE 7

DECLARATION

As a below named inventor, I hereby declare that:

I am a United States citizen. My residence address is set forth below beneath my name.

The below named inventors are the original, first and joint inventors of the subject matter which is claimed and for which a patent is sought on the invention entitled **METHOD FOR EXECUTING TRUSTED PATH COMMANDS**, the specification of which was filed as **Serial No. 479,666**, on February 13, 1990. I have reviewed and understand the contents of the above identified specification, including the claims as amended by a Preliminary Amendment filed herewith. I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, § 1.56(a).

I hereby direct that all correspondence and telephone calls be addressed to D. C. Toedt, Esq., Regis. No. 31,144, Arnold, White & Durkee, P.O. Box 4433, Houston, Texas 77210, (713) 787-1400.

I hereby declare that all statements made of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

_____ May 16 1990
Mary Ellen Zurko   (date)
230 Nashua Road, Groton MA 04250

_____ 17 May 90
Andrew Halstead Mason   (date)
61 Baxter Road, Hollis NH 03049

_____
Thomas Andrew Casey, Jr.   (date)
194 School Street, Westwood MA 02090

_____ 17 May 1990
Paul Douglas Sawyer   (date)
16 Governor Peabody Rd, Billerica MA 01821

_____ 5/18/90
Morrie Gasser   (date)
11 Golden Hills Road, Saugus MA 01906

_____ 5/16/90
Leslie Richard Kendall   (date)
5 Goodwin Road, Glouchester MA 01930

_____ 5/16/90
Judith Shelhorse Hall   (date)
47 Indian Ridge Road, Sudbury MA 01776

_____ 16 May 90
Steven B. Lipner
6 Midland Road, Wellesley MA 02181

_____ 16 May 1990
Clifford Earl Kahn   (date)
15 Weld St., Unit 26, Framingham MA 01701

DECLARATION

As a below named inventor, I hereby declare that:

I am a United States citizen. My residence address is set forth below beneath my name.

The below named inventors are the original, first and joint inventors of the subject matter which is claimed and for which a patent is sought on the invention entitled **METHOD FOR EXECUTING TRUSTED PATH COMMANDS**, the specification of which was filed as **Serial No. 479,666**, on February 13, 1990. I have reviewed and understand the contents of the above identified specification, including the claims as amended by a Preliminary Amendment filed herewith. I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, § 1.56(a).

I hereby direct that all correspondence and telephone calls be addressed to D. C. Toedt, Esq., Regis. No. 31,144, Arnold, White & Durkee, P.O. Box 4433, Houston, Texas 77210, (713) 787-1400.

I hereby declare that all statements made of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Mary Ellen Zurko   (date)
230 Nashua Road, Groton MA 04250

Thomas Andrew Casey, Jr.   5-17-90 (date)
194 School Street, Westwood MA 02090

Morrie Gasser   (date)
11 Golden Hills Road, Saugus MA 01906

Judith Shelhorse Hall   (date)
47 Indian Ridge Road, Sudbury MA 01776

Clifford Earl Kahn   (date)
15 Weld St., Unit 26, Framingham MA 01701

Andrew Halstead Mason   (date)
61 Baxter Road, Hollis NH 03049

Paul Douglas Sawyer   (date)
16 Governor Peabody Rd, Billerica MA 01821

Leslie Richard Kendall   (date)
5 Goodwin Road, Glouchester MA 01930

Steven B. Lipner
6 Midland Road, Wellesley MA 02181